Un exercice complexe : le compresseur Huffman

Pierre Pompidor

Le compresseur Huffman

Compresser des fichiers sans perte d'information

Au lieu que chaque caractère soit encodé sur un multiple d'octets associer un encodage binaire inférieur, égal ou supérieur à 8 bits suivant la **fréquence** du caractère dans le texte

Le compresseur Huffman

Compresser des fichiers sans perte d'information

Au lieu que chaque caractère soit encodé sur un multiple d'octets associer un encodage binaire inférieur, égal ou supérieur à 8 bits suivant la **fréquence** du caractère dans le texte

Plusieurs étapes

- compter le nombre d'occurrences de chaque caractère
- créer l'arbre binaire permettant de définir les nouveaux encodages
- parcourir cet arbre de la racine aux feuilles pour les définir
- récréer le contenu du fichier suivant ce nouvel encodage (en associant un index permettant de le décompresser)
- (et créer le décompresseur)

Comptage du nombre d'occurrences de chaque caractère

Lecture globale du fichier (si c'est possible)

```
if len(sys.argv) > 1
   and os.access(sys.argv[1],os.R_OK) :
  with open(sys.argv[1]) as fd :
    texte = fd.read()
    tabOccurrences = compterOccurrences(texte)
```

Comptage du nombre d'occurrences de chaque caractère

Lecture globale du fichier (si c'est possible)

```
if len(sys.argv) > 1
   and os.access(sys.argv[1],os.R_OK) :
  with open(sys.argv[1]) as fd :
    texte = fd.read()
    tabOccurrences = compterOccurrences(texte)
```

Création d'une liste de 256 éléments initialisés à 0

```
occurrences = [0 for _ in range(0, 256)]
```

Comptage du nombre d'occurrences de chaque caractère

Lecture globale du fichier (si c'est possible)

```
if len(sys.argv) > 1
   and os.access(sys.argv[1],os.R_OK) :
  with open(sys.argv[1]) as fd :
    texte = fd.read()
    tabOccurrences = compterOccurrences(texte)
```

Création d'une liste de 256 éléments initialisés à 0

```
occurrences = [0 for _ in range(0, 256)]
```

Imaginons que le contenu du fichier soit aaaaabbbbcccdde

ord() pour avoir le code ascii des caractères : ord('a') renvoie 97

Création de l'arbre binaire

Principe de l'algorithme

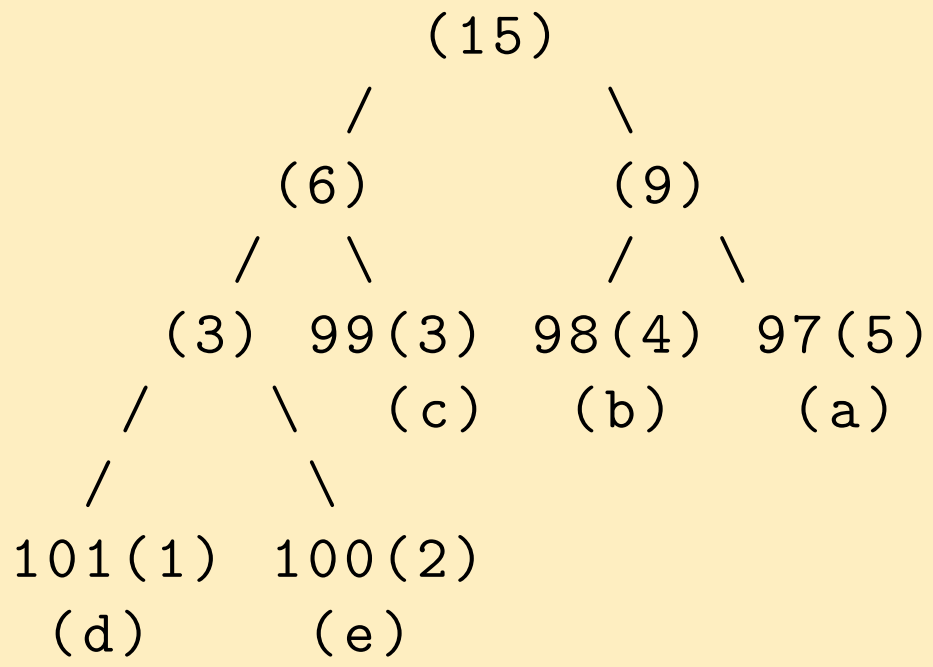
Contruire un arbre binaire à partir de ses feuilles
en affiliant les noeuds non ancore utilisés
associés au plus petits nombres d'occurrences

Création de l'arbre binaire

Principe de l'algorithme

Contruire un arbre binaire à partir de ses feuilles
en affiliant les noeuds non ancore utilisés
associés au plus petits nombres d'occurrences

Pour l'exemple "aaaaabbbbcccdde"



Création de l'arbre binaire - suite

Structuration d'un noeud

(nombre d'occurrences cumulées, code ascii, fils gauche, fils droit)

Création de l'arbre binaire - suite

Structuration d'un noeud

(nombre d'occurrences cumulées, code ascii, fils gauche, fils droit)

```
def creerArbre(occurrences):
    noeuds = [(occurrences[i], i, (), ())
        for i in range(0,256) if occurrences[i]>0]
    while len(noeuds) > 1 :
        noeuds = sorted(noeuds, key=lambda x:x[0])
        noeudMin1 = noeuds.pop(0) # pop() retire
        noeudMin2 = noeuds.pop(0)
        ...   # A COMPLETER
    return noeuds
```

Parcours de l'arbre binaire et création des encodages

Les nouveaux codes sont donnés par le parcours de l'arbre

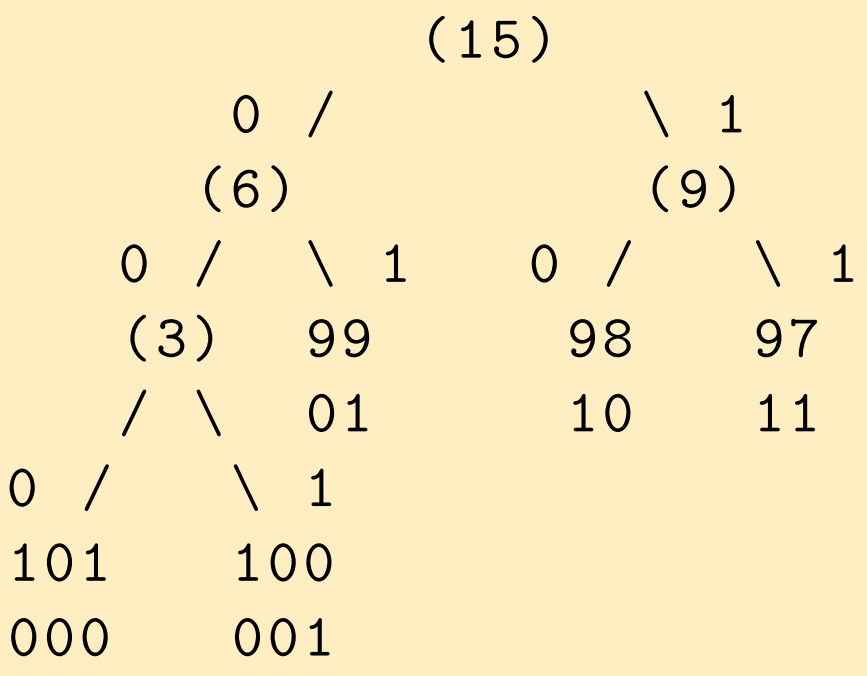
de la racine aux feuilles en "colorant" les arêtes avec 0 ou 1

Parcours de l'arbre binaire et création des encodages

Les nouveaux codes sont donnés par le parcours de l'arbre

de la racine aux feuilles en "colorant" les arêtes avec 0 ou 1

Les nouveaux codes sont donnés par le parcours de l'arbre



Parcours de l'arbre binaire et création des encodages - suite

La fonction de parcours est doublement récursive

en paramètre, une chaîne de caractères va agglomérer
des 0 (rappel sur le premier sous-arbre) ou des 1 (ou sur l'autre)

Parcours de l'arbre binaire et création des encodages - suite

La fonction de parcours est doublement récursive

en paramètre, une chaîne de caractères va agglomérer
des 0 (rappel sur le premier sous-arbre) ou des 1 (ou sur l'autre)

Le dictionnaire qui mémorise les nouveau codes

```
encodageBinaire = {}
```

Parcours de l'arbre binaire et création des encodages - suite

La fonction de parcours est doublement récursive

en paramètre, une chaîne de caractères va agglomérer
des 0 (rappel sur le premier sous-arbre) ou des 1 (ou sur l'autre)

Le dictionnaire qui mémorise les nouveau codes

```
encodageBinaire = {}
```

Le squelette de la fonction

```
def creerEncodageBinaire(sousArbre, encodage):
    if sousArbre[2] != () : # il y a deux fils
        # A COMPLETER
    else :
        encodageBinaire[sousArbre[1]] = encodage
```

Création du fichier compressé

Utilisation d'opérateurs binaires

décalage vers la gauche : «
ou binaire : |

Création du fichier compressé

Utilisation d'opérateurs binaires

décalage vers la gauche : «
ou binaire : |

Décalage vers la gauche (exemple)

```
b = 1              # ou 0b00000001
print(b << 1)  # 2 ou 0b00000010
```

Création du fichier compressé

Utilisation d'opérateurs binaires

décalage vers la gauche : «
ou binaire : |

Décalage vers la gauche (exemple)

```
b = 1              # ou 0b00000001
print(b << 1)  # 2 ou 0b00000010
```

Le ou binaire (exemple)

```
b = 1              # ou 0b00000001
print(b | 2)   # 3 ou 0b00000011
```

Création du fichier compressé (suite)

```
fr = open(sys.argv[1]+".huf",'wb')
```

Création du fichier compressé (suite)

```
fr = open(sys.argv[1]+".huf",'wb')
```

Création d'un nouvel octet tous les 8 bits

```
nbBits = 0
byte = 0  # ou 0b00000000
for caractere in texte :
  for b in encodageBinaire[ord(lettre)] :
    nbBits += 1
    # A COMPLETER
    if nbBits == 8 :
      fr.write(byte.to_bytes(1,byteorder='big'))
      byte = 0b00000000; nbBits = 0
```

Création du fichier compressé (suite)

```
fr = open(sys.argv[1]+".huf",'wb')
```

Création d'un nouvel octet tous les 8 bits

```
nbBits = 0
byte = 0  # ou 0b00000000
for caractere in texte :
  for b in encodageBinaire[ord(lettre)] :
    nbBits += 1
    # A COMPLETER
    if nbBits == 8 :
      fr.write(byte.to_bytes(1,byteorder='big'))
      byte = 0b00000000; nbBits = 0
```

Pour visualiser le contenu du fichier : xxd -b fichier.huf

Pour finir le compresseur

Il faudrait

- Ajouter l'arbre en index du fichier compressé
- Créer le décompresseur

Pour finir le compresseur

Il faudrait

- Ajouter l'arbre en index du fichier compressé
- Créer le décompresseur

Pour aller encore plus loin

un archiveur serait à portée de mains !